JOURNAL OF COMBINATORIAL THEORY (A) 19, 192–199 (1975)

A Class of Nonlinear de Bruijn Cycles

HAROLD FREDRICKSEN

Institute for Defense Analyses, Princeton, New Jersey 08540

Communicated by Marshall Hall, Jr.

Received December 26, 1974

We describe an algorithm for determining a nonlinear de Bruijn sequence of length 2^n requiring only $3n$ bits of storage. We then find a collection of cross-joins which can be applied to the sequence to produce a large class of related de Bruijn cycles. We give some ideas on the implementation of the generation of these sequences and some warnings about their possible deficiencies.

1. MAXIMAL SEQUENCES

A problem which was of interest in the mid-1940's was to determine an oriented circular pattern of 2^n bits with the property that each of the n-bit patterns occurs exactly once in the pattern. The existence of a solution was shown in [1, 2]. The method employed was to traverse an oriented linear graph. This method was not constructive and no explicit solutions were given. In [2] the total number of solutions was shown to be $2^{2^{n-1}-n}$.

In [3] an explicit solution to the problem is given which makes use of primitive polynomials over GF [2]. It is this class of solution which is best known. Of the total number of solutions $\varphi(2^n - 1)/n$ are of this "linear" form. φ is Euler's φ-function.

A "nonlinear" solution was given by several authors [4–6]. Their solution was to start with n zeros and append a one as the next bit of the sequence as long as the n-tuple formed had not previously appeared, otherwise append a zero. This algorithm was generalized in [7] to form all de Bruijn cycles (as they came to be known). It should be noted that the solutions as presented in the above references were not restricted to the binary case.

Another generalization of the nonlinear solution was given in [8, 9]. Each of these authors used a graph-theoretic approach to generate the de Bruijn cycles. Other solutions result from obtaining a solution for n and "mapping up" to a solution for $n + 1$ [10–12].

2. The Lexicographically Least de Bruijn Cycle

The nonlinear de Bruijn cycle given above is called the lexicographically least de Bruijn cycle (lldBs) [13] because it is the first de Bruijn cycle in a dictionary where one precedes zero.

We can represent the sequence as the output of a nonlinear feedback shift register. The feedback function of the register is the function $f(x_1x_2 \cdots x_n)$. We restrict our attention to branchless feedbacks; thus we impose the condition that $f(x_1x_2 \cdots x_n) = x_n + g(x_1 \cdots x_{n-1})$ [7]. The feedback $f = x_n + g$ will generate a partition of the 2^n points of the space so that every point has a unique predecessor as well as successor. When all of the 2^n points are on a single cycle we call the cycle a de Bruijn cycle.

In [13] the truth table of g for the lldBs is examined and shown to have $Z(n) - 1$ ones, where

$$Z(n) = 1/n \sum_{d|n} \varphi(d)\, 2^{n/d}$$

is the number of cycles generated by the feedback $f = x_n$, i.e., $g \equiv 0$. The ones in the truth table of g correspond to the positions on the sequence where the n-tuple $0x_{n-1} \cdots x_1 \to x_{n-1} \cdots x_1 1$ and $1x_{n-1} \cdots x_1 \to x_{n-1} \cdots x_1 0$. Note that $Z(n) - 1$ ones is the smallest number of ones any truth table can contain and produce a de Bruijn cycle. This since a single binary change in the truth table of g can only change the number of cycles produced by f by one. (The single change in g produces two changes in f by the restriction to branchless feedbacks.) Thus if we want to produce a de Bruijn (one) cycle from the $Z(n)$ cycles determined by the feedback $g \equiv 0$ we must introduce at least $Z(n) - 1$ ones into g. Further, [13] shows some positions of g which will be one when f produces the lldBs. Another paper [14] further studies the problem of determining the truth table of g.

We note that the original algorithm for the lldBs requires that we know the entire sequence to the point we are now considering. In effect, to produce the sequence we require storage of all 2^n bits of the sequence. Using the ideas of [13, 14] we can determine the ones in the truth table of g and these positions can be stored. This requires storage of $Z(n) - 1$ numbers of $n - 1$ bits and since $Z(n) \sim 2^n/n$ we require $n - 1/n2^n$ bits for this implementation. The algorithm we describe below makes a considerable improvement on this storage requirement.

In [13] we list the $Z(n)$ cycles produced by $g \equiv 0$ and determine how to append each of these cycles to the collection of previously amassed cycles. When all elements of a cycle have been included in the listing of the lldBs we say the cycle is exhausted. The cycles are exhausted in order by the number of ones on the cycles. In particular, the first cycle exhausted is

the cycle (1) which would have n ones if thought of as filling an n-stage register by itself. The next exhausted cycle is (011 $\cdots$ 1) containing $n-1$ ones. In general, the next cycle exhausted is the cycle containing the largest element on all cycles not already exhausted. The significance of this is as follows. Each cycle has one entry point from the previously amassed collection of cycles. That is, if we call the weight of a cycle generated by g to be the number of ones on the cycle (when viewed as an n-long cycle), then each cycle has only one entry point from a cycle with weight one fewer. It may have several entry points from cycles with weight one more.

The key point to note here is what the successors of an n-tuple tell us about the sequence previously described. There are four situations to consider;

(a) $0x_{n-1} \cdots x_1 \to x_{n-1} \cdots x_1 0$,
(b) $0x_{n-1} \cdots x_1 \to x_{n-1} \cdots x_1 1$,
(c) $1x_{n-1} \cdots x_1 \to x_{n-1} \cdots x_1 0$,

and

(d) $1x_{n-1} \cdots x_1 \to x_{n-1} \cdots x_1 1$.

Situations (a) and (d) are similar. In each case we are on one of the cycles generated by the feedback $g \equiv 0$ and the successor chosen by the algorithm is the next element on the respective cycle. Situation (a) occurring further says that previously (d) ($1x_{n-1} \cdots x_1 \to x_{n-1} \cdots x_1 1$) appeared on the cycle generated, else the 1 successor would have been chosen; (d) occurring says that (b) has not previously appeared; and (c) occurring implies that (b) has previously appeared. Then we note that returns (c) from cycles with weight one more to cycles with weight one smaller can occur only after the jump (b) from cycles with weight one fewer to cycles with weight one more. Repeated application of this idea shows that each cycle has a single entry from a cycle with one fewer one.

When it is time to add a cycle we can say which element of the cycle will be entered from an element off the cycle. The element chosen on each cycle is the odd part of the largest element on the cycle [13, 15]. This follows because the cycles are entered (and exhausted) in turn according as they contain the largest number on a cycle not yet joined to the previously amassed collection. Then the algorithm is as follows.

ALGORITHM [15]. (0) $\beta_0 = (0, 0, \ldots, 0)$ the starting n-tuple of all zeros.

From $\beta_i = (b_i b_{i+1} \cdots b_{i+n-1})$ we produce $\beta_{i+1} = (b_{i+1} \cdots b_{i+n})$.

(1) Form $\beta_i^* = (b_{i+1} \cdots b_{i+n-1} 1)$.

(2) Consider all cyclic shifts of β_i^* to find the maximum element M_i on the "cycle" β_i^*, $M_i = (b_s \cdots b_{i+n-1}1b_{i+1} \cdots b_{s-1})$.

(3) If $b_{i+1} = b_{i+2} = \cdots = b_{s-1} = 0$, then $\beta_{i+1} = (b_{i+1}, \ldots, b_{i+n-1}, \bar{b}_i)$, otherwise $\beta_{i+1} = (b_{i+1} \cdots b_{i+n-1}b_i)$.

The algorithm is verified by noting that if $b_{i+1} = \cdots = b_{s-1} = 0$ then $b_{i+1} \cdots b_{i+n-1}1$ is the odd part of the largest number on its cycle and is the element to be mapped into from the element with one fewer one. Then $g(b_{i+1}, \ldots, b_{i+n-1}) = 1$ in the truth table. If $b_i = 0$ the action of g is to add the new cycle β_i^* to the collection of already amassed cycles. If $b_i = 1$ the action of g is to return from β_i^* to the cycle from which β_i^* was entered originally.

The storage required for the above algorithm is n bits for the current position on the cycle β_i, n bits for cycling β_i^*, and n bits for the current highest value on the cycle β_i^*. The algorithm will require n units of time to produce the next bit every time that there is a cycle change. Whenever there is no cycle change (when b_i is cycled around uncomplemented), less than n units of time will be required. Thus only $2/n$ of the time will n units of time be required for the next bit.

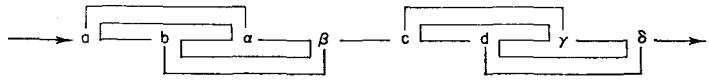
Another feature of the last algorithm is that it can be started at any starting point $b_ib_{i+1} \cdots b_{i+n-1}$ because we do not need to keep track of the old values on the cycle.

3. Relatives of the 11dBs

By a process of cross-joining it is possible to convert one de Bruijn cycle to another. All that is required is to find two vectors a and b such that their respective alternate successors α and β are positioned with β falling between a and α and α falling between b and β. Then forcing $a \to \alpha$ and $b \to \beta$ will produce a different de Bruijn cycle than the original.

If we have a double cross-join *that does not cross itself*, then changes in both pairs may be made while preserving the de Bruijn property.

Lemma 1. *If two pairs of cross-joins $a\alpha$; $b\beta$ and $c\gamma$; $d\delta$ are in the order $ab\ \alpha\beta\ cd\ \gamma\delta$ on a de Bruijn cycle, then mapping $a \to \alpha$, $b \to \beta$, $c \to \gamma$, and $d \to \delta$ will produce another de Bruijn cycle.*



We of course could allow c to appear before any or all of a, b, α, β. There are many other positions which are obviously equivalent to what we describe.

LEMMA 1′. *If we can find k cross-joined pairs $a_i, \alpha_i ; b_i, \beta_i$, $i = 1, 2,..., k$ such that no two of them intersect we can make the cross-joins in any of the k pairs independently to produce a set of 2^k de Bruijn cycles.*

The reason for this is that the change of successors on any subset of this set of cross-joined pairs, will not alter the relative order of the vectors in any of the remaining cross-joined pairs. Therefore they will still be cross-joined after the change of successors has been made.

LEMMA 2. *The pairs $a_i, \alpha_i ; b_i, \beta_i = 2^i, 2^{i+1}; 2^{n-2-i}, 2^{n-1-i} + 1$ are independent cross-join pairs in the* lldBs *for $i = 0, 1,..., [n - 3/2]$.*

Proof. $2^j + 1$ is the odd part of the largest number on its cycle only when $j = 1, 2,..., [n - 1/2]$. Thus $2^{j-1} \to 2^j + 1$ only for these values of j. Then $2^j \to 2^{j+1}$ for $j = [n - 1/2] + 1,..., n - 2$ and $2^{n-1} \to 0$. Also since $2^j \to 2^{j+1} + 1$ for $j = 0, 1,..., [n - 3/2]$ and the algorithm for the sequence places one before zero, $2^{j+1} + 1$ procedes 2^{j+1} for those values of j. The relative order of the powers of two and the numbers $2^i + 1$ when n is even is thus

$$\overbrace{1, 2 + 1,..., 2^{n-1} + 1}, \overbrace{2, 2^2 + 1,..., 2^{n-2} + 1}, \overbrace{2^2,...,...},$$

$$\overbrace{2^{n/2-1}, 2^{n/2} + 1,..., 2^{n/2}}, 2^{n/2+1}, 2^{n/2+2},..., 2^{n-2}, 2^{n-1}0.$$

When n is odd a similar sequence is derived which ends with

$$2^{(n-1)/2} \to 2^{(n+1)/2} \to 2^{(n+3)/2} \to \cdots \to 2^{n-2} \to 2^{n-1} \to 0.$$

Then we have established the order for the elements $a_i\alpha_i b_i\beta_i$.

LEMMA 3. *The pairs*

$$a_i, \alpha_i ; b_i, \beta_i = 2^i - 1, 2^{i+1} - 2; \sum_{j=1}^{i} 2^{n-1-j}, 1 + \sum_{j=1}^{i} 2^{n-j}$$

are independent cross-join pairs in the lldBs *for $i = 1, 2,..., n - 2$.*

The proof of Lemma 3 follows in a very similar fashion to that of Lemma 2 will be omitted. We also note that outside of where the sets of Lemmas 2 and 3 coincide, namely in the cross-join pair 1, 2; 2^{n-2}, $2^{n-1}+1$, there is no interference between the sets since all of the action of Lemma 2 follows the element 2 and all of the action of Lemma 3 precedes the element 2.

An alternative means of determining cross-join pairs is to follow the methods outlined in [10, 16, 17]. These papers give methods for finding all of the cross-join pairs obtainable from the cyles formed from the feedback function $g \equiv 0$. It may be that these results may allow us extra cross-join pairs.

We are able to obtain the result

LEMMA 2′. *The pairs* a_i, α_i ; $b_i, \beta_i = 2^{n-1} - 1 - 2^i$, $2^n - 1 - 2^{i-1}$; $2^{n-1} - 1 - 2^{n-2-i}$, $2^n - 1 - 2^{n-3-i}$ *are independent cross-join pairs in the* lldBs *for* $i = 0, 1, \ldots, [n - 3/2]$.

The interference patterns with the cross-joins in Lemma 3 can be shown to be independent of the cross-joins in Lemma 2′. That is, when any cross-joined pair is applied the relative orders of the elements of the other cross-joined pairs are unaffected. We get an additional group of $[n - 3/2]$ independent cross-join pairs from Lemma 2′ and thus we have:

THEOREM. *From the* lldBs *we can by cross-join pairs generate a collection of* 2^{2n-5} *nonlinear de Bruijn cycles.*

Also, other cross-joins in the prime n case are possible. For example, $n = 7$ allows 12 cross-join pairs but we only get 9 from the theorem.

As an example of these ideas we let $n = 5$. The lldBs is

0 1 3 7 15 31 30 29 27 23 14 28 25 19 6 13
26 21 11 22 12 24 17 2 5 10 20 9 18 4 8 16.

The Lemma 2 cross-join pairs are 1, 2; 8, 17 and 2, 4; 4, 9. (The Lemma 2′ cross-joins are 13, 27; 11, 23 and 14, 29; 7, 14.) The Lemma 3 cross-join pairs are 1, 2; 8, 17; 3, 6; 12, 25 and 7, 14; 14, 29. If we change g so that $g(2) \Rightarrow 0$, $g(4) \Rightarrow 1$, $g(7) \Rightarrow 0$ and $g(14) = 1$ we get the new de Bruijn cycle

0 1 3 7 14 29 27 23 15 31 30 28 25 19 6 13
26 21 11 22 12 24 17 2 4 9 18 5 10 20 8 16.

4. Implementation

If we wish to generate any of these 2^{2n-5} nonlinear de Bruijn cycles the algorithm will closely follow the algorithm [15] described above with the following modifications. We create a vector c of length n for the cross-join pairs of Lemma 2. $c_i = c_{n-2-i} = 1$ if we wish to employ the cross-join pair 2^i, 2^{i+1}; 2^{n-2-i}, $2^{n-1-i} + 1$. Likewise, create a vector c' for the cross-joins of Lemma 2'. We let $c_i' = 1$ if we wish to apply the cross-join pair $2^{n-1} - 1 - 2^i$, $2^n - 1 - 2^{i-1}$; $2^{n-1} - 1 - 2^{n-2-i}$, $2^n - 1 - 2^{n-3-i}$. Then step (3) of the algorithm is changed to

(3') If

$$b_{i+1} = b_{i+2} = \cdots = b_{s-1} = 0$$

then $\beta_{i+1} = (b_{i+1}, \ldots, b_{i+n-1}, \bar{b}_i + c_j)$; otherwise $\beta_{i+1} = (b_{i+1}, \ldots, b_i + c_j)$ whenever $b_i, \ldots, b_{i+n-1}$ contains a single 1 in position $b_{i+n-1-j}$ and c_j is the jth element of the vector c. Similarly, we can apply the vector c'. Whenever $b_i, \ldots, b_{i+n-1}$ is of the form $(01 \cdots 101 \cdots 1)$ with the second zero in position $b_{i+n-1+k}$ we add c_k' to $\bar{b}_i$ or b_i.

For the cross-join pairs of Lemma 3 we create a vector D of length $n - 2$ and if $b_i, \ldots, b_{i+n-1}$ is of the form

$$00 \cdots 0\overbrace{1 \cdots 1}^{k} \quad \text{or} \quad \overbrace{01 \cdots 1}^{k}0 \cdots 0$$

we also implement a change in (3) of the algorithm to accommodate the cross-join pairs of Lemma 3. Thus we add the entry D_k from the vector D to $\bar{b}_i$ or b_i in the two cases. Thus the complete implementation uses $<6n$ bits of storage.

5. Some Caveats

The truth table for the lldBs has $Z(n) - 1$ ones. While the auto-correlation of the sequence is small for out-of-phase values τ not a multiple of n, when τ is a multiple of n we get high correlation values. If we define the out-of-phase auto-correlation at distance τ to be $c(\tau)$ = agreements of lldBs with lldBs shifted by τ bits, then

$$c(n) = 2(2^{n-1} - (Z(n) - 1)) \sim 2(2^{n-1} - (2^n/n)) = (n - 2)\, 2^n/n.$$

Note also the same correlation values hold for all of the 2^{2n-5} relatives of the lldBs since for each of the cross-joins we changed one position of g to 1 and one position of g to 0.

This poor correlation that we observe is of course a global phenomenon. It may very well be that locally some or perhaps even all of the sequences have good correlation properties for some long stretch of bits.

The lldBs has another poor quality which it shares with certain linear de Bruijn cycles. It is known for trinomial rules for generation of maximal sequences that there may be some local roughness, perhaps 55%–60% of the bits in a short stretch might be one. For $n = 59$, the first million bits of the lldBs are over 90% ones. It is not known what kind of mixing is performed by the cross-joins described above, but it seems clear that this figure of 90% ones will be substantially reduced.

REFERENCES

1. I. J. GOOD, Normal recurring decimals, *London Math. Soc.* **21** (1946), 167–169.
2. N. G. DE BRUIJN, A combinatorial problem, *Nederl. Akad. Wetensch. Proc.* **49** (1946), 758–764.
3. D. REES, Note on a paper by I. J. Good, *J. London Math. Soc.* **21** (1946), 169–172.
4. M. H. MARTIN, A problem in arrangements, *Bull. Amer. Math. Soc.* **40** (1934), 859–864.
5. R. LESSARD, Cycles of n-digit binary integers—solution to prob. 4385, *Amer. Math. Monthly* **58** (October 1951), 573–575.
6. L. R. FORD, A cyclic arrangement of m-tuples, Report No. P-1071, RAND Corp., 1957.
7. S. W. GOLOMB, "Shift-Register Sequences," p. 133, Holden–Day, San Francisco, 1967.
8. F. J. MOWLE, Relation between P_n cycles and stable feedback shift registers, *IEEE Trans. Electronic Computers* **EC-15** (1966), 375–378.
9. D. KNUTH, "The Art of Computer Programming," Vol. I., "Fundamental Algorithms," Addison Wesley Reading, Mass., 1968, p. 375.
10. K. B. MAGELBY, The synthesis of non-linear feedback shift registers, Tech. Rept. 6207-1, pp. 60–62, Stanford Electronic Labs., 1963.
11. E. B. LEACH, Regular sequences and frequency distributions, *Proc. Amer. Math. Soc.* **11** (1960), 566–574.
12. A. LEMPEL, On a homomorphism of the de Bruijn graph and its applications to the design of feedback shift registers, *IEEE Transactions Computers* **C-19** (1970), 1204–1209.
13. H. FREDRICKSEN, The lexicographically least de Bruijn cycle, *J. Combinatorial Theory* **9** (1970), 1–5.
14. S. MOSSIGE, Constructive theorems for the truth table of the Ford sequence, *J. Combinatorial Theory, Ser. A* **11** (1971), 106–110.
15. H. FREDRICKSEN, Generation of the Ford sequence of length 2^n, n large, *J. Combinatorial Theory, Ser. A* **12** (1972), 153–154.
16. E. J. VAN LANTSCHOOT, Double adjacencies between cycles of a circulating shift register, *IEEE Trans. Electronic Computers* **C-22** (1973), 944–954.
17. J. MYKKELTVEIT, Generating and counting the double adjacencies in a pure cycling shift register, *IEEE Trans. Electronic Computers*, **C-24** (1975), 299–304.